Remarks

The present amendment responds to the Official Action dated March 29, 2004. The Official Action objected to the drawings. The Official Action rejected claims 1-12 under 35 U.S.C. 102(b) based on Schwaller U.S. Patent No. 5,838,919 ("Schwaller"), Autrey U.S. Patent No. 5,774,695 ("Autrey"), Chirashnya U.S. Patent No. 6,560,720 ("Chirashnya")and Cidon U.S. Patent No. 6,269,330 ("Cidon"), individually. These grounds of rejection are addressed below following a brief discussion of the present invention to provide context. Claims 1, 7 and 9 have been amended to be more clear and distinct. Claims 1-12 are presently pending.

The Present Invention

A system according to an aspect of the present invention includes an endpoint and an emulator module associated with the endpoint. The emulator module includes at least one finite state machine (FSM) for modeling traffic flows to be emulated. The traffic flows being emulated such that a relatively small constant number of processes can maintain simultaneous emulation of a relatively large number of traffic flows.

The emulator module preferably includes an event scheduler containing an event queue, and an emulation manager that manages the flows that are emulated by the emulator module.

The emulation manager preferably includes an FSM that maintains the status of the emulation, as well as an FSM for each flow. The FSMs may be created according to scripts, with each FSM suitably being created according to a script corresponding to a half flow associated with the endpoint. A half flow is a flow from a source to a destination, or from a destination to a source. Emulation is suitably driven by events that are initiated or reacted to by FSMs. An FSM may react differently to the same event, depending on the state of the FSM, and the occurrence of an event may cause the FSM to change states or to create another event.

Each packet in a flow is preferably created and sent in response to a triggering event. An emulator preferably performs emulation of a traffic flow by assembling events in an event queue and processing events until an exit state is reached. The use of an emulation module using finite state machines to model traffic flows allows for a relatively large number of traffic flows to be emulated using a relatively small constant number of processes, and allows for a large number of flows to be modeled without unduly increasing processor overhead.

The Objection to the Drawings

The Official Action object to the drawings on the ground that Figs. 1 and 5 should be labeled with a designation such as "Prior Art" on the ground that only old material is illustrated by these figures. This objection is respectfully traversed. Figs. 1 and 5 do not illustrate prior art. Fig. 1 illustrates a system embodying the present invention. The system includes endpoints modeling traffic flows using finite state machines, as described in the discussion. Fig. 5 illustrates a topology in which endpoints according to the present invention may be employed. Figs. 1 and 5 do not illustrate the emulator modules and finite state machines associated with the endpoints, because Figs. 1 and 5 show a general level of detail at which specific illustration of the emulator modules and finite state machines is not appropriate. Figs. 1 and 5 do, however, illustrate the present invention and not prior art. The objection to the drawings should therefore be withdrawn.

The Art Rejections

All of the art rejections hinge on the application of either Schwaller, Autrey, Chirashnya or Cidon, standing alone. As addressed in greater detail below, the cited references do not support the Official Action's reading of them and the rejections based thereupon should be reconsidered and withdrawn. Further, the Applicant does not acquiesce in the analysis of the cited references made by the Official Action and respectfully traverses the Official Action's analysis underlying its rejections.

The Official Action rejected claims 1-12 under 35 U.S.C. 102(b) as anticipated by Schwaller, Autrey, Chirashnya and Cidon, individually. In light of the present amendments to claims 1 and 7, this ground of rejection is respectfully traversed.

Claim 1, as amended, claims an endpoint and an emulator module associated with the endpoint. The emulator module comprises at least one finite state machine for modeling traffic flows to be emulated. The traffic flows are emulated such that a constant number of processes, the number of processes being small relative to the number of traffic flows that can be emulated, can maintain simultaneous emulation of a relatively large number of traffic flows. The number of processes required for emulation is independent of the number of traffic flows to be emulated. Neither Schwaller, Autrey, Chirashnya nor Cidon teaches these features.

Schwaller does not teach the use of finite state machines, and does not teach emulation of traffic flows such that a relatively small constant number of processes can maintain simultaneous emulation of a relatively large number of traffic flows, with the number of processes required for emulation being independent of the number of traffic flows to be emulated. A finite state machine, or FSM, is an entity that can be set to different states, typically by the occurrence of events, and responds differently to events depending on the state that the FSM is in when the event occurs. The use of an emulator module using at least one finite state machine for modeling traffic flows, as claimed by claim 1, allows for event driven modeling of traffic flows, with flexibility in responding to events. An FSM may respond to events or sequences of events in different ways, and may respond differently to a repetition of an event or a sequence of events. The reason for this is that an event may set an FSM to a new state, and once the FSM is in the new state, it may react differently if the event is repeated. Emulating traffic flows such that a relatively small constant number of processes can emulate a relatively large number of traffic flows, with the number of processes required for emulation being independent of the number of traffic flows to be emulated, allows for good scalability for modeling of traffic flows. If a relatively small constant number of processes is used, and the number of processes required for emulation is independent of the number of flows to be emulated, an increase in the number of flows to be emulated does not add substantial overhead. Because the number of processes can remain constant, emulation of a larger number of flows requires the same processor or operating system overhead, or only a small increase in overhead, as compared with emulation of a smaller number of flows. By contrast, if simultaneous traffic flows are modeled by concurrently running processes or threads, processor or operating system overhead may increase substantially as the number of flows to be modeled increases, because addition of flows to be emulated requires addition of simultaneously running processes.

Schwaller does not manage emulation so as to emulate simultaneous traffic flows using a relatively small constant number of processes, with the number of processes required for emulation being independent of the number of traffic flows to be emulated. See Schwaller, col. 31, lines 45-53, which describes the concurrent handling of multiple scripts as accomplished, for example, through the use of multi-threading in OS/2 ™. The use of multiple processes or threads increases the overhead imposed as the number of processes or threads increases, and limits the number of simultaneous process or threads that can be managed. The present invention, as claimed by claim 1, as amended, is able to overcome these difficulties and achieve emulation of multiple simultaneous traffic flows while requiring little processor or operating system overhead. Claim 1, as amended, therefore defines over Schwaller.

Similarly, Autrey does not teach an emulation module comprising at least one finite state machine for modeling traffic flows to be emulated, the traffic flows being emulated such that a relatively small constant number of processes can maintain simultaneous emulation of a relatively large number of traffic flows, with the number of processes required for emulation being independent of the number of traffic flows to be emulated. Autrey teaches an interface gateway and method of connecting an emulator to a network. Autrey provides for facilities for selecting the type of emulator to which connection is to be made. See Autrey, col. 10, lines 19-32, describing selection of a socket based emulator or a "normal" emulator. However, Autrey does not appear to offer details of any emulator to which connection may be made, and does not describe an emulator module employing a finite state machine, or the use of a finite state machine to emulate traffic flows such that a relatively small constant number of processes can maintain simultaneous emulation of a relatively large number of traffic flows, with the number of processes required for emulation being independent of the number of traffic flows to be emulated. Claim 1, as amended, therefore defines over Autrey.

Similarly, Chirashnya does not teach an emulation module comprising at least one finite state machine for modeling traffic flows to be emulated, with the traffic flows being emulated such that a relatively small constant number of processes can maintain simultaneous emulation of a relatively large number of traffic flows, with the number of processes required for emulation being independent of the number of traffic flows to be emulated. Chirashnya teaches injection of errors into a network using an error injector node which is isolated, or fenced, from the normal program and message flow in the network. The error injector node is not explicitly described as maintaining simultaneous emulation of a large number of traffic flows using a relatively small constant number of processes. Injection of errors, as performed by Chirashnya, is likely to constitute a relatively restricted form of traffic and is not likely to call for the same degree of complexity demanded by emulation of more general traffic flows. Therefore, Chirashnya is not likely to face the same need to manage simultaneous traffic flows as does the present invention as claimed by claim 1. The invention as claimed by claim 1, on the other hand, may be used to model a comprehensive traffic pattern, which may easily comprise multiple simultaneous traffic flows, and may advantageously manage the traffic flows so as to avoid creating excessive overhead. Claim 1, as amended, therefore defines over Chirashnya.

Similarly, Cidon does not teach an emulation module comprising at least one finite state machine for modeling traffic flows to be emulated, the traffic flows being emulated such that a relatively small constant number of processes can maintain simultaneous emulation of a relatively large number of traffic flows, with the number of processes required for emulation being independent of the number of traffic flows to be emulated. Cidon teaches testing of networks, and teaches the use of a traffic generator in the course of such testing. The traffic generator of Cidon preferably comprises software processes and/or subroutines that run on a host. The processes may operate substantially concurrently. Cidon also notes that it is possible to implement a similar generator in hardware. See Cidon, col. 12, lines 15-22. Cidon thus teaches the modeling of traffic through the use of multiple concurrent processes, whether operating in software or hardware, not the use of a finite state machine to model traffic flows such that a relatively small constant number of processes can maintain emulation of a relatively large number of traffic flows, with the number of processes required for emulation being independent of the number of traffic flows to be emulated. Claim 1, as amended, therefore defines over Cidon and the rest of the cited art and should be allowed.

Claim 7, as amended, claims providing an emulator module comprising providing at least one finite state machine for modeling traffic flows to be emulated, the traffic flows being emulated such that a constant number of processes, the number of processes being small relative to the number of traffic flows that can be emulated, can maintain simultaneous emulation of a relatively large number of traffic flows, the number of processes required for emulation being independent of the number of traffic flows to be emulated. Claim 7 further claims modeling traffic flows with said at least one finite state machine. For the reasons stated above with respect to claim 1, neither Schwaller, Autrey, Chirashnya nor Cidon teaches these limitations. Claim 7, as amended, therefore defines over the cited art and should be allowed.

Conclusion

All of the presently pending claims, as amended, appearing to define over the applied references, withdrawal of the present rejection and prompt allowance are requested.

Respectfully submitted,

[illegible]

Joseph B. Agusta
Reg. No. 52,547
Priest & Goldstein, PLLC
5015 Southpark Drive, Suite 230
Durham, NC 27713-7736
(919) 806-1600